LOGIC PROGRAM COMPARISON METHOD FOR VERIFYING A COMPUTER PROGRAM IN RELATION TO A SYSTEM SPECIFICATION

BACKGROUND OF THE INVENTION

The invention relates to a process for verifying the correctness of an implementation by comparing it with a system specification on which the implementation is based, each represented by logic programs.

To build a system, such as a computer system which consists of hardware and software, the specifications are first prepared and, based on them, an actual system is implemented. The specifications, which describe what a designer wishes the implementation to satisfy, are generally written in a style that is unambiguous yet easy to read. On the other hand, the implementation, which describe software and hardware which embody the specifications, is written in a style which permits the details of how a software and hardware component will solve a given problem.

Verification means the analysis of the implementation to determine whether all parts of the specifications are satisfied by the implementation. Verification is vital in building an error-free system according to the designer's intents.

During development of system hardware or software, it is desirable to determine whether the implementation meets all parts of the specifications. Verification during development significantly enhances system reliability and eliminates the need for backtrack processes during development.

The implementation that satisfies all the conditions expressed in the specification is said to satisfy the specification. As a system becomes larger and more complicated, it is very important, during system development, to ensure that the specification is satisfied.

There are several description formats for the specification, and the designer uses one of the formats depending upon the types of conditions to be verified. For asynchronous systems such as communication protocols, a specification format in which conditions are represented based on the timing of events is desirable. A specification description language for this purpose, such as temporal logic, is available. For synchronous systems containing many hardware units, it is desirable that specifications express conditions concerning functionality in terms of the relationship between the input and output of hardware and software components. The present invention relates to the latter type of input-output correctness verification.

There is a method for verifying the output correctness of an implementation with respect to a specification. In this method, both the implementation and the specification are translated to finite state machine (FSM) representations (hereafter called FSM) and then these two FSM representations are compared. The FSM representation eliminates syntactic differences (such as variable names) between the specification and the implementation, and provides a common representation of the semantic content of the specification and the implementation, facilitating comparison.

The FSM itself can be represented using boolean expressions. Since the FSMs are often very large, the boolean expressions can in turn be represented using binary decision diagrams (BDDs).

Not only are BDDs able to represent the internal contents of FSMs, but a set of algorithms exist which allow fast processing of large boolean expressions.

To make the comparison of two FSMs meaningful, it is usually assumed that both FSMs accept the same set of possible input values.

When two FSMs are compared, it is required that, in each FSM state, both machines will generate the same outputs for identical inputs. The two machines start in their respective initial states. That is, the current state of each machine is assumed to be the initial state. And, for the current state, a check is made to ensure that the corresponding outputs of the two machines are identical for all possible input values. This process is repeated for all the possible states. When the check is made for all the possible states, the identity of two FSMs can be determined.

Using a BDD based comparison of FSMs results in extremely short computer execution time. However, the BDD representation used in this method has limited expressiveness, and unable to take direct advantage of patterns in the state space. This means that the compactness of BDD representation depends on the ability to find patterns in the boolean expressions. In addition, the efficiency of BDD representations are largely affected by the order of the variables occurring in the boolean expressions, yet it is extremely difficult to determine the order of variables. Because of this, it is difficult to do effective verification in the existing FSM based method.

Another drawback of existing FSM based verification methods is that only strictly finite state machines can be compared. In the verification of an actual system, it is often desirable to work with parameterized specifications and implementation descriptions. In that case, however, existing methods are not efficient because some elements of the system are left as unspecified variables. For example, when comparing a stack implementation description against a stack specification, the details of the stack contents are left as unspecified parameters. Existing methods do not handle this case.

In view of the foregoing, it is the main object of the present invention to provide a method to solve the problem of the prior art. More specifically, it is an object of this invention to provide a method whereby verification is performed with parameterized logic programs and to provide an efficient logic program comparison method.

SUMMARY OF THE INVENTION

The present invention has the following effects: Two logic programs entered by the input step are converted to the first and second FSM descriptions by a converting step. The converting step determines the data types of each program, converts each program to the completed form, expands the procedure calls in each program to procedure bodies, permutes the resulting procedure dies based on the variable order, and replaces each representation in the program by a unique code. Thus, canonical FSMs which are suitable for verification may be obtained, making comparison between FSMs easy. Then, the comparison step determines whether there exists an equivalence between enumerated states, between input values, and between output values, and determines whether the descriptions produce respective output values deemed equal for all respective inputs deemed equal for all respective states deemed equal, and outputs the result of the comparison. Thus, the equality relation need not be strict equality but need only satisfy a given relation, making it possible to verify parameterized logic programs.

According to the invention the converted contents to be compared are restricted to generic queries, making the comparison more efficient.